Error Mining with Suspicion Trees: Seeing the Forest for the Trees

Shashi Narayan, Claire Gardent

Error Mining with Suspicion Trees: Seeing the Forest for the Trees

Shashi Narayan[1] Claire Gardent[2]
(1) Université de Lorraine/LORIA, Nancy, France
(2) CNRS/LORIA, Nancy, France
shashi.narayan@loria.fr, claire.gardent@loria.fr

Abstract

In recent years, error mining approaches have been proposed to identify the most likely sources of errors in symbolic parsers and generators. However the techniques used generate a flat list of suspicious forms ranked by decreasing order of suspicion. We introduce a novel algorithm that structures the output of error mining into a tree (called, suspicion tree) highlighting the relationships between suspicious forms. We illustrate the impact of our approach by applying it to detect and analyse the most likely sources of failure in surface realisation; and we show how the suspicion tree built by our algorithm helps presenting the errors identified by error mining in a linguistically meaningful way thus providing better support for error analysis. The right frontier of the tree highlights the relative importance of the main error cases while the subtrees of a node indicate how a given error case divides into smaller more specific cases.

Title and Abstract in Hindi

संदेह वृक्षों के साथ त्रुटि खननः वृक्षों की खोज जंगल में

हाल के वर्षों मे, प्रतीकात्मक पद-व्याखक तथा पद-उत्पादक की गलतियों के संभावित श्रोतों की पहचान के लिये कई त्रुटि खनन तकनीकें प्रस्तावित की गयी हैं। हालांकि प्रस्तावित तकनीकें संदेहात्मक उदाहरणों को एक साधारण सूची में उनके घटते संदेह के क्रम में प्रस्तुत करते हैं। हम यहाँ एक नयी तकनीक प्रस्तुत कर रहे हैं जो त्रुटि खनन के परिणामों को एक वृक्ष (संदेह वृक्ष) के रूप में गठित कर संदेहात्मक उदाहरणों के बीच के संबंधों को दिखलाता है। हम अपने तकनीक की उपयोगिता पद-उत्पादक की संभावित असफलताओं के श्रोतों को ढूँढने तथा विश्लेषण करके प्रस्तुत करते हैं। हम यह दिखाते हैं कि कैसे हमारे तकनीक द्वारा निर्मित संदेह वृक्ष त्रुटि खनन द्वारा पहचानीत गलतियों को भाषाशास्त्र के दृष्तिकोण से काफी अर्थपुर्ण तरीके से प्रस्तुत कर त्रुटि-विश्लेषण में काफी मददगार साबित होता है। संदेह वृक्ष का दाहिना सीमांत मुख्य त्रुटिपुर्ण उदाहरणों को उनके तुलनात्मक महत्ता के साथ प्रस्तुत करता है जबकि संदेह वृक्ष की शाखायें दिखातीं हैं कि कैसे एक त्रुटिपुर्ण उदाहरण छोटे-छोटे त्रुटिपुर्ण विशिष्ट उदाहरणों में बँटता है।

Keywords: Error Mining, Generation.

Keywords in Hindi: त्रुटि खनन, पद-उत्पादक.

1 Introduction

In recent years, error mining approaches have been proposed to identify the most likely sources of errors (called, *Suspicious Forms*) in symbolic parsers and generators. (van Noord, 2004) initiated error mining on parsing results with a very simple approach computing the parsability rate of each n-grams in a very large corpus. The parsability rate of an n-gram $w_i \dots w_n$ is the ratio $P(w_i \dots w_n) = \frac{C(w_i \dots w_n | OK)}{C(w_i \dots w_n)}$ where $C(w_i \dots w_n)$ is the number of sentences in which the n-gram $w_i \dots w_n$ occurs and $C(w_i \dots w_n \mid OK)$, the number of sentences containing $w_i \dots w_n$ which could be parsed. In other words, the parsability rate of an n-gram is the proportion of sentences in which this n-gram occurs and for which parsing succeeds. An n-gram then, is a suspicious form if it has a low parsability rate.

(van Noord, 2004)'s approach was extended and refined in (Sagot and de la Clergerie, 2006), (de Kok et al., 2009) and (Gardent and Narayan, 2012) as follows. (Sagot and de la Clergerie, 2006) defines a suspicion rate for n-grams which takes into account the number of occurrences of a given word form and iteratively defines the suspicion rate of each word form in a sentence based on the suspicion rate of this word form in the corpus. Further, (de Kok et al., 2009) extends this iterative error mining to n-grams of words and POS tags of arbitrary length. And (Gardent and Narayan, 2012) extends (van Noord, 2004)'s approach to mine for suspicious subtrees rather than n-grams.

An important limitation shared by all these error mining approaches is that their output is a flat list of suspicious forms ranked by decreasing order of suspicion. There is no clear overview of how the various suspicious forms interact and as a result, the linguist must "hop" from one error case to another instead of focusing on improving sets of related error cases. In short, the output of these error mining approaches lacks structure thereby making it difficult to handle errors in a linguistically meaningful way.

To overcome this shortcoming, we propose an algorithm which structures the output of error mining into a *suspicion tree* making explicit both the ranking of the main distinct error cases and their subcases. The suspicion tree is a binary tree structure whose internal nodes are labelled with suspicious forms and whose leaf nodes represent the clusters of error mined data grouped according to the suspicious forms characterizing their elements. Like in a decision tree, each cluster in the suspicion tree is characterized by the set of attributes (suspicious forms) labelling its ancestors; and the tree itself represents a disjunction of mutually exclusive error cases.

We illustrate the impact of our error mining algorithm on error analysis by applying it to detect and analyse the most likely sources of failure in a surface realiser developed to participate in the Surface Realisation Shared Task (Belz et al., 2011); and we show how this error mining algorithm permits improving the surface realiser.

The paper is structured as follows. We start (Section 2) by introducing our error mining algorithm. In essence, this algorithm adapts (Quinlan, 1986)'s ID3 algorithm to build a suspicion tree such that the clusters obtained group together sets of input data that share similar sources of failure (called *suspicious forms*); and the attributes labelling these clusters are the suspicious forms indicating which are these most likely causes of failure. In Section 3, we show how this error mining algorithm helps improving a surface realiser executed on the input dependency trees provided by the Surface Realisation (SR) Task challenge. Section 4 concludes with pointers for further research.

2 Building Suspicion Trees

In this section, we introduce the *suspicion tree algorithm* and discuss its complexity.

2.1 The Suspicion Tree Algorithm

As mentioned above, our error mining algorithm resembles (Quinlan, 1986)'s ID3 decision tree learning algorithm, in that it recursively partitions the data by first, selecting the attribute (here, a suspicious form) that best divides the data into more homogeneous subsets (*attribute selection*) and second, using this attribute to split the data into two subsets, a subset containing that attribute and a subset excluding that attribute (*dataset division*).

In what follows, we define the metric used to recursively select a suspicious form and partition the data, namely the *Suspicion Score* metric. We specify the termination conditions. We illustrate by means of examples how suspicion trees help structure the output of error mining. And we contrast the suspicion tree algorithm with (Quinlan, 1986)'s ID3 decision tree learning algorithm.

The Suspicion Score Metrics. Let $\mathbb{D}$ be the dataset to be error mined and $\mathbb{F}$ be the set of attributes used to partition the data. Here, $\mathbb{D}$ is a set of dependency trees provided for the Surface Realisation Task by the Generation Challenge; and $\mathbb{F}$ is the set of subtrees of $\mathbb{D}$ whose frequency is above a given threshold. Following (Gardent and Narayan, 2012), we use a complete and efficient Hybrid Tree Miner algorithm (Chi et al., 2004), to compute the set of subtrees that are present in $\mathbb{D}$.

Let $\mathbb{D}$ be divided into two disjoint sets: PASS (*P*) is the set of instances $t^P \in \mathbb{D}$ for which the processing system (e.g., a parser or a generator) succeeds; and FAIL (*F*) is the set of instances $t^F \in \mathbb{D}$ for which the system fails. Given these two sets, the **suspicion score** $S_{score}(f)$ of a form $f \in \mathbb{F}$ is then defined as follows:

$$S_{score}(f) = \frac{1}{2}(\text{Fail}(f) * \ln count(f) + \text{Pass}(\neg f) * \ln count(\neg f))$$

Intuitively, this metric captures the degree to which a form is associated with failure: it is high whenever a form f is often present in data associated with failure (high *F(ail)-Suspicion*, $\text{Fail}(f)$) and/or when it is often absent in data associated with success (high *P(ass)-Suspicion*, $\text{Pass}(\neg f)$).

The **F-Suspicion rate of** f is defined as the proportion of cases where f occurs in an instance t for which the processing system fails:

$$\text{Fail}(f) = \frac{count(f|\text{FAIL})}{count(f)}$$

$count(f)$ is the number of instances containing f and $count(f|\text{FAIL})$ is the number of instances containing f for which processing failed.

Conversely, the **P-Suspicion rate of a form** f is defined as the proportion of cases not containing f and for which processing succeeds ($count(\neg f)$ is the number of instances where f is absent and $count(\neg f|\text{PASS})$ is the number of instances not containing f for which processing succeeds):

$$\text{Pass}(\neg f) = \frac{count(\neg f|\text{PASS})}{count(\neg f)}$$

Attribute Selection, Dataset Division and Termination. The suspicion tree algorithm selects at each step of the tree building process, the form f with highest suspicion score i.e. the form such that, in the current dataset, most instances that contain f fail to be processed; and most instances that excludes f lead to successful processing.

Based on this selected f, the current dataset is divided into two subsets: the set of instances which contain f and the set of instances which exclude f.

The form selection and dataset division process are called recursively on the new subsets until (i) the obtained set of instances is fully homogeneous (all instances in that set lead to either successful or unsuccessful processing); (ii) all forms have been processed; or (iii) the depth upper bound is reached (see below).

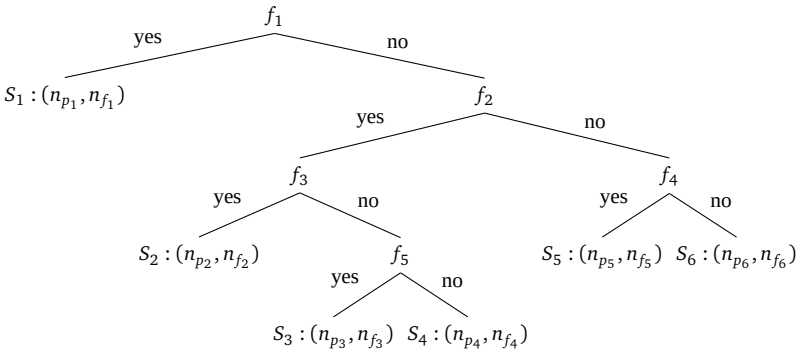


Figure 1: An example *Suspicion Tree*. Internal nodes are labeled with suspicious forms and leaves indicate the number of instances in the current data set S_i for which processing succeeds (n_{p_i}); and for which processing fails (n_{f_i}). When the sources of errors are clearly identifiable, n_{p_i} will be low, n_{f_i} will be high and the rightmost leaf (f_4) will have a low n_{f_i}.

Example. Figure 1 shows an abstract suspicion tree which illustrates how suspicion trees help structuring the output of error mining. The right frontier highlights the relative importance of the main distinct error cases while subtrees indicate how a given error case divides into smaller more specific cases. The branches of the tree also indicate the combinations of forms that frequently cooccur in failure cases.

More specifically, the root f_1 of this *suspicion tree* is the most suspicious form present in the corpus $\mathbb{D}$. Starting from the root, following the edges with label "no" (the *right-frontier* of the tree i.e., f_1, f_2 and f_4) yields the ranked list of suspicious forms present in $\mathbb{D}$ by decreasing order of suspicion. Following branches yields datasets labeled with sets (conjunctions) of suspicious forms. For example, the set S_2 with n_{p_2} of pass instances and n_{f_2} of failed instances has f_2 and f_3 as their top ranked suspicious forms. The *suspicion tree* also displays the relative ranking of the suspicious forms. For example, the set $(S_2 \cup S_3 \cup S_4)$ has f_2 as its most suspicious form, and f_3, f_5 as its next two most suspicious forms. Moreover, most of the instances in S_1, S_4 and S_5 fail because of a single form namely, f_1, f_2 and f_4 respectively.

Suspicion tree algorithm vs. ID3 algorithm. There are two main differences between (Quinlan, 1986)'s ID3 decision tree learning algorithm and the suspicion tree construction algorithm.

First, the suspicion tree construction algorithm allows for stronger pruning and termination conditions – in this way, only the most relevant suspicious forms are displayed thereby facilitating error analysis.

Second, attribute selection is determined not by the information gain (IG) but by the suspicion score

(SS) metrics. Recall that the information gain[1] metrics aims to identify the attributes which lead to more homogeneous classes. In the present case, the classes are either PASS (the inputs for which generation succeeds) or FAIL (the inputs for which generation fails). Thus the IG metrics will indifferently seek to identify attributes which predominantly associate either with a FAIL or with a PASS. There is no preference for either the FAIL or the PASS class. For error mining however, what is needed is to identify attributes which predominantly associate with the FAIL class. That is, we need a metric which permits identifying attributes which leads to classes that are homogeneous in terms of FAIL instances rather than homogeneous in terms of either FAIL or PASS instances. The example shown in Figure 2 illustrates the difference.

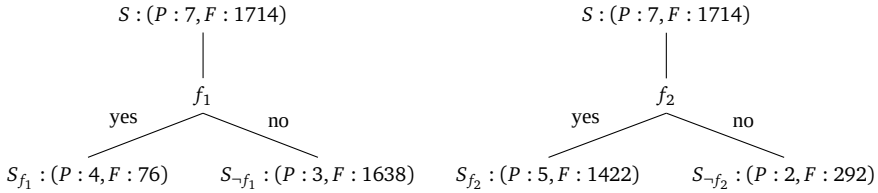


Figure 2: Attribute Selection using Information Gain (Left) and Suspicion Score (Right). While IG selects f_1, an attribute which associate 76 times with generation failure, SS selects f_2, an attribute which associates 1422 times with generation failure.

In this example, we apply the IG and the SS metrics to the same input data, a set containing 7 inputs associated with generation success and 1714 inputs associated with generation failure. While SS selects f_2, an attribute which associates 1422 times with generation failure, IG selects f_1, an attribute which associate only 76 times with generation failure. In this case, the information gain metrics incorrectly select f_1 because its absence from the input, yields a numerically very homogeneous class in terms of generation failure. Indeed, the information gain of f_1 is close to but higher than the information gain of f_2 because the resultant subsets S_{f_i} and $S_{\neg f_i}$ are treated equally while computing the information gain.

2.2 Complexity Analysis and Extensions

Let n and m be the size of the dataset $\mathbb{D}$ and of the form set $\mathbb{F}$ respectively. Then, in the worst case, the suspicion tree will be of depth $O(log\ n)$ with $O(n)$ nodes. Each node chooses a suspicious form out of $O(m)$ forms. Thus the worst computational complexity for building the suspicion tree is $O(m\ n\ log\ n)$. But on average, the algorithm described in Section 2.1 performs much faster than this. The worst case happens when the forms used to classify the corpus into PASS and FAIL are not very discriminant i.e., when all suspicious forms are equally probable.

The algorithm for building the suspicion tree is directly proportional to the size of the set $\mathbb{F}$. Since $|\mathbb{F}|$ can be very large, this can be problematic. Indeed, in the error mining on sentences for parsing systems proposed in (Sagot and de la Clergerie, 2006), the authors indicate that, in order to remain computationally tractable, the approach must be restricted to n-grams of smaller size (unigrams and bigrams). The problem is accrued of course when considering tree shaped suspicious forms (Gardent and Narayan, 2012). To abate this issue we propose two extensions to prune the suspicion tree.

[1]Information gain (IG) is defined as $IG = H(S) - ((|S_{f_i}|/|S|) * H(S_{f_i}) + (|S_{\neg f_i}|/|S|) * H(S_{\neg f_i}))$ where $H(X)$ is the entropy of set X. (Quinlan, 1986)

First, we reduce the form space $\mathbb{F}$. Following a suggestion from (de Kok et al., 2009), instead of considering all possible forms, we only consider those forms whose frequency is above a given threshold. We also account for *suspicion sharing* (i.e., the sharing of suspicion by several overlapping forms) by only considering a larger suspicious form if its suspicion rate is larger than the suspicion rate of all smaller forms it contains. These two extensions reduce the form space significantly and allow for an efficient building of the suspicion tree. To enumerate with these extensions, we use a complete and efficient algorithm described in (Gardent and Narayan, 2012).

Second, we constrain the depth of the suspicion tree. Because error mining is a cyclic process, building the complete suspicion tree is usually unnecessary. The quantity of information processed in each cycle depends on the user but in general, the linguist will focus on the top suspicious forms, use these to improve the generator and rerun error mining on the improved results. The faster the error mining step is, the better this is for this development cycle. Considering this, we added an extra constraint over the depth of the suspicion tree. This depth limit permits pruning the suspicion tree and a faster improvement cycle. In our experiments, we used a depth limit of 10.

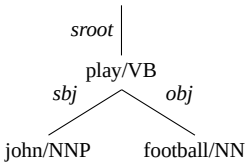
With these extensions, the enumeration process of suspicious forms takes 10-15 minutes for a dataset consisting of 123,523 trees. Building a suspicion tree for the same dataset takes about one minute.

3 Applying the Suspicion Tree Algorithm to Generation Data

We now report on an experiment we did using the suspicion tree algorithm described in the preceding section to detect and classify the most likely causes of failure when running a surface realiser on the Surface Realisation (SR) Task data. We first describe the experimental setup (Section 3.1). We then illustrate by means of examples, how suspicion trees better support error analysis than ranked lists proposed by previous error mining approaches (Section 3.2). Finally (Section 3.3), we discuss the improvements in surface realisation obtained by fixing the errors identified using error mining.

3.1 Experimental Setup

Dataset The dataset to be error mined is the set of shallow dependency trees (Figure 3) provided by the SR Task organisers and used as input for surface realisation. These trees are unordered syntactic dependency trees whose edges are labelled with dependency relations and whose nodes are labelled with lemmas and part of speech (POS) categories. In this paper, we represent these trees by an n-tuple with the root node of the tree as its first element followed by $(n-1)$ elements representing its dependent subtrees. Dependency relations are lowered to the corresponding daughter node.



word (play, (john), (football))

POS (VB, (NNP), (NN))

dep (*sroot*, (*sbj*), (*obj*))

word/**POS** (play/VB, (john/NNP), (football/NN))

***dep*-POS** (*sroot*-VB, (*sbj*-NNP), (*obj*-NN))

Figure 3: An example shallow dependency tree from the SR Task and the corresponding representations used in this paper. Our error mining algorithm considers as suspicious forms, subtrees labelled with arbitrary conjunctions of lemmas (word), part-of-speech tags (POS), dependency relations (*dep*).

To facilitate error mining, we proceed in an incremental way and examine dependency trees in the SR data that correspond to NP and Sentences of increasing size. Here we report on error mining performed on NP-type dependency trees of sizes 4 (NP-4), 6 (NP-6) and all (NP-ALL), and S-type dependency trees of sizes 6 (S-6), 8 (S-8) and all (S-ALL) (where the size refer to the number of nodes/lemmas in the tree). The data used for generation is preprocessed whereby named entities and hyphenated words are grouped into a single word and punctuation is removed so as to first focus on lexical and grammatical issues.

Attributes The attributes used to partition the SR data are suspicious trees i.e., subtrees of the SR dependency trees whose frequency is above a given threshold. Following (Gardent and Narayan, 2012), we allow for various views on errors by mining for forms labelled with lemmas only (word); with parts of speech (POS); with dependency relations (*dep*); with lemmas and parts of speech (word/POS); and with dependency relations and parts of speech (*dep*-POS) (cf. Figure 3).

Generation System The system to be tested is the symbolic Surface Realiser described in (Narayan and Gardent, 2012). We ran this surface realiser on the SR input data and separately stored the input dependency trees for which generation succeeded (PASS) and the input dependency trees for which generation failed (FAIL). We then removed from the failed data, those cases where generation failed either because a word was missing in the lexicon or because a grammar rule was missing but required by the lexicon and the input data. These cases can easily be detected using the generation system and thus do not need to be handled by error mining.

Error Mining We iterate several times between error mining and performance improvement and applied the suspicion tree algorithm to both the NP and the S data[2].

3.2 Error Analysis using Suspicion Trees

We now show by means of examples how the suspicion tree algorithm helps support error analysis. We start by showing how the overall structure of the suspicion tree (right frontier and subtrees) improves upon ranked lists when analysing the data. We then go on to show how subtrees in the suspicion tree permit differentiating between forms that are suspicious in all contexts and require a single correction; forms that are suspicious in all contexts but require several corrections; and forms that are suspicious in some but not all contexts.

3.2.1 Suspicion Trees vs. Ranked Lists

Figure 4 shows a top fragment of the suspicion tree obtained by error mining on NP-4. The node labels in this tree describe suspicious forms with part-of-speech information only.

In that tree, the right frontier indicates that the main distinct suspicious forms are, in that order:

1. Possessive NPs (POSS is the part of speech tag assigned to possessive *'s*[3])
The suspicious form (POSS) points to a mismatch between the representation of genitive NPs (e.g., *Oakland's thief*) in the SR Task data and in the grammar. While our generator expects the representation of '*Oakland's thief*' to be (thief/NN, ('s/POSS, (oakland/NNP))), the structure provided by the SR Task is (thief/NN, (oakland/NNP, ('s/POSS))). Hence whenever a possessive appears in the input data, generation fails. This is in line with (Rajkumar et al., 2011)'s finding that the logical

[2]Iteration stops either when the results are perfect (perfect coverage and perfect BLEU score) or when the trees fail to be discriminative enough (low number of FAIL instances associated with the suspicion tree leaves). So far, the latter situation did not occur and we are still using the suspicion tree to identify the main sources of errors for the remaining error cases.

[3]In fact, the part of speech tag assigned to possessive *'s* in the SR data is POS not POSS. We renamed it to avoid confusion with POS as an abbreviation for part-of-speech.

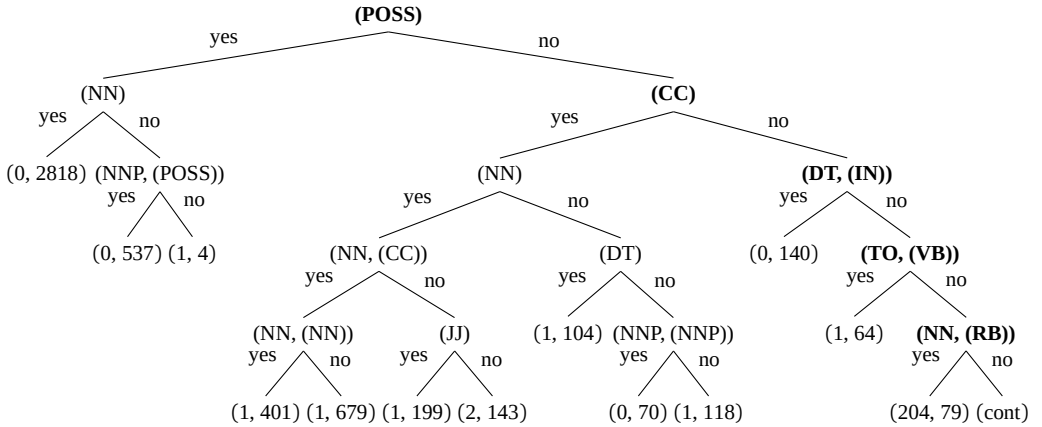


Figure 4: Suspicion Tree for Generation from the NP-4 data. Nodes are labelled with dependency subtrees with POS information. The leaves (p, f) represent the cluster with PASS (p) and FAIL (f) instances.

forms expected by their system for possessives differed from the shared task inputs. To correct these cases, we implemented a rewrite rule that converts the SR representation of possessive NPs to conform with the format expected by our realiser.

2. NPs with coordination (CC with daughter node NN)

The second top right frontier node unveils a bug (conflicting feature values) in the grammar trees associated with NP conjunction (e.g., *Europe and the U.S.*) which made all sentences containing an NP conjunction fail.

3. Determiners (DT) dominating a preposition (IN)

As we shall see below, this points to a discrepancy between the SR part of speech tag assigned to words like '*some*' in '*some of the audience*' and the part of speech tag expected by our generator. While in the SR data, such occurrences are labelled as determiners (DT), our generator expects these to be tagged as pronouns (PRP).

4. The complementizer *to* (TO) dominating a verb (VB)

As discussed below, this points to cases where the infinitival verb is a noun modifier and the input structure provided by the SR Task differs from that expected by our realiser.

5. Nouns (NN) dominating an adverb (RB)

This points to a discrepancy between the SR part of speech tag assigned to words like 'alone' in '*real estate alone*' and the part of speech tag expected by our generator. While in the SR data, such occurrences are labelled as adverbs (RB), our generator expects these to be tagged as adjectives (JJ).

In addition, for each node n on the right frontier, the subtree dominated by the yes-branch of n gives further information about the more specific forms that are subcases of the suspicious form labelling n.

The suspicion tree gives a structured view of how the various suspicious forms relate. In comparison, the ranked lists produced by previous work are flat structures which may fail to adequately display these information. For instance, applying (Gardent and Narayan, 2012)'s error mining algorithm to the data used to produce the tree shown in Figure 4 yields the list shown in Figure 5. Contrary to the suspicion tree shown in Figure 4, this list fails to highlight the main culprits and

1. **(POSS)**
2. (NNP, (POSS))
3. **(CC)**
4. (NN, (POSS))
5. (NN, (NNP, (POSS)))
6. (NN, (NN, (POSS)))
7. (NN, (CC))
8. (NNP, (NNP), (POSS))
9. (NN,(NNP,(NNP),(POSS)))
10. (NN, (NNP, (NNP)))
11. (CC, (JJ))
12. (JJ, (CC))
13. (NNP, (NNP, (POSS)))
14. (NN, (NN), (POSS))
15. **(DT, (IN))**
16. (JJ, (CC, (JJ)))
17. (NN, (CC), (NN))
18. (NN, (NNP), (POSS))
19. **(TO, (VB))**
20. (NN,(NNP,(POSS)),(NNP))
21. (NN, (NNP))
22. (NNP, (NNP))
23. (NN, (NN))
24. (NNP)
25. (NN)
26. (NN, (NNP), (NNP))
27. (VB)
28. **(NN, (RB))**
29. (PRP)
30. (DT)

Figure 5: Ranked list of suspicious forms for Generation from the NP-4 data.

the relations between the various suspicious forms. Thus the 5 main distinct suspects identified by the right frontier of the suspicion tree appears as 1st, 3rd, 15th, 19th and 28th in the ranked list. Furthermore, while subcases of the main suspects are grouped in the yes-branch of these suspects in the suspicion tree, in the ranked list, they appear freely interspersed throughout. For example, suspicious forms involving the two main suspects in the suspicion tree approach (POSS and CC part-of-speech tags) are scattered throughout the list rather than grouped under the first two right frontier nodes respectively.

Also the stronger pruning conditions used for building the suspicion tree restrict the branch exploration as soon as homogeneous clusters are achieved. For a given dataset, it only explores those suspicious forms which are good enough to identify the problems causing the failure in that dataset. For example the data containing the suspicious form (POSS) is explored with 3 suspicious forms (POSS), (NN) and (NNP, (POSS)) in the suspicion tree shown in Figure 4 whereas in the ranked list shown in Figure 5, there are 11 suspicious forms associated with (POSS). In general, the number of forms displayed by the suspicion tree algorithm is much less than that of the ranked list ones thereby giving a clearer picture of the main culprits and of their subcases at each stage in the error mining/grammar debugging cycle.

3.2.2 Reading error types off the tree structure

For each node n labelled with suspicious form f_n in a suspicion tree, the subtree dominated by n gives detailed information about the possible contexts/causes for f_n. In what follows, we show how the suspicion tree algorithm permits distinguishing between three main types of suspicious forms namely, forms that are suspicious in all contexts and require a single correction; forms that are suspicious in all contexts but require several corrections; and forms that are suspicious in some but not all contexts.

Forms that are suspicious independently of context and require a single correction. When a suspicious form always leads to failure, the node labelled with that suspicious form has no subtree thereby indicating that all configurations including that suspicious form lead to generation failure independent of context.

Such cases are illustrated in Figure 6 which show two views (one with part of speech tag only, the other with words and parts of speech tags) of the suspicion tree obtained after addressing the two main causes of errors identified in the previous section. That is, a rewrite rule was applied to convert the SR representation of possessive NPs to conform with the format expected by our

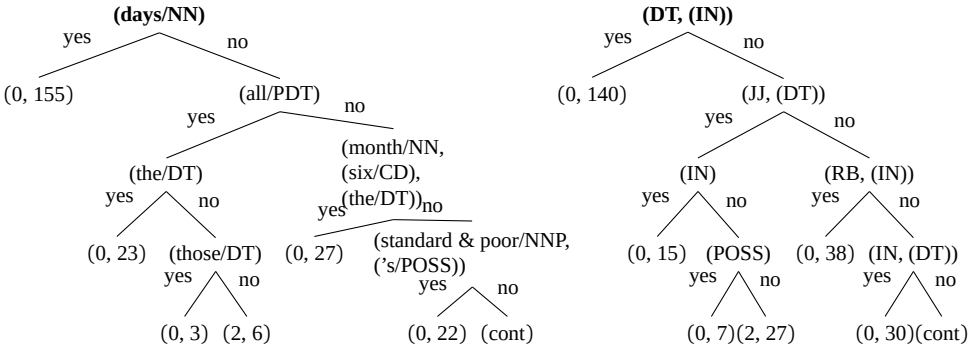


Figure 6: Suspicion Tree for (word/POS) (left) and (POS) (right) for Generation from the NP-4 data (after fixing genitive and coordination cases).

realiser ((POSS) suspicious form); and the grammar was corrected to generate for NP coordination ((CC) suspicious form).

In each of these two trees, the yes-branch of the root node has no subtree indicating that all input trees containing either the word form '*days*' with part of speech tag NN (days/NN); or a determiner dominating a preposition ((DT,(IN))) lead to generation failure.

The root node (days/NN) of the suspicion tree shown on the left of Figure 6 points to a problem in the lexicon. Although days/NN is present in the lexicon, it is not associated with the correct TAG family. We modified the entries corresponding to (days/NN) in the lexicon to solve this problem.

As mentioned above, the root node (DT, (IN)) of the suspicion tree shown on the right in Figure 6 points to a part-of-speech tagging problem in the SR Data. Words like '*some*' or '*all*' followed by a preposition (e.g., *some of the audience*, *all of fiscal 1990*, *those in other industries*) are assigned the determiner part of speech tag (DT) where our generator expects a pronoun (PRP) part-of-speech tag. To correct these cases, we implemented a rewrite rule that maps DT to PRP in the above specified context.

As these two examples illustrate, using different views (forms labelled with part of speech tags only *vs.* forms labelled with words and parts of speech) on the same data may help identifying problems at different levels. Both suspicion trees in Figure 6 are built for generation from same NP-4 dataset. The leftmost tree (suspicious forms labelled with both lemma and part of speech information) helps identifying problems in the lexicon whereas the rightmost tree (suspicious forms labelled with parts of speech only) points to problems in the input data.

Forms that are suspicious independent of context but require several corrections. It may be that a given form is almost always suspicious but that it occurs in different linguistic contexts requiring different corrections. In such cases, the suspicion tree will highlight these contexts. The root of the tree shown in Figure 7 is a case in point. The suspicious form (*im*-VB) describes subtrees whose head is related to a verb by the *im* dependency relation i.e., *infinitival verbs*. The subtrees (of the yes-branch) of that root further describe several syntactic configurations which are suspect and contain an infinitival verb. The node labelled with (*oprd*-TO) points to subcases where the infinitival verb is the complement of a control (1a[i]) or a raising verb (1a[ii]). The node labelled with (*im*-VB, (*prd*-JJ)) points to a subcase of that case namely that of an infinitival verb which is

the complement of a control or a raising verb and subcategories for an adjectival complement e.g., (1b). Finally, the node labelled with (*nmod*-TO, (*im*-VB)) points to cases where the infinitival verb is a noun modifier (1c).

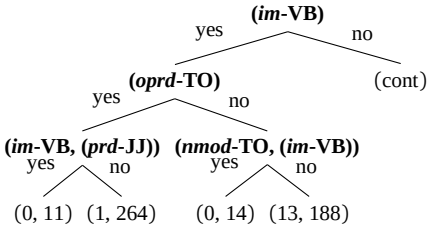


Figure 7: Suspicion Tree (*dep*-POS) for Generation from the S-6 data.

(1) a. (*oprd*-TO)

i *He will try to assuage the fears about finances.*
(try/VB, (*oprd*-to/TO, (*im*-assuage/VB))

ii *Many of the morning session winners turned out to be losers.*
(turn/VB, (*oprd*-to/TO, (*im*-be/VB, (*prd*-loser/NN)))

b. (*im*-VB, (*prd*-JJ))
Amex expects to be fully operational by tomorrow.
(expect/VB, (*oprd*-to/TO, (*im*-be/VB, (*prd*-operational/JJ)))

c. (*nmod*-TO, (*im*-VB))
The ability to trade without too much difficulty has steadily deteriorated.
(ability/NN, (*nmod*-to/TO, (*im*-trade/VB))

(2) a. (IN, (CD))
the end of 1991
(end/NN, (the/DT), (of/IN, (1991/CD)))

b. (CD, (IN))
one of the authors
(one/CD, (of/IN, (author/NN, (the/DT))))

c. (CD, (CD))
Nov. 1 , 1997
(1/CD, (1997/CD), (Nov./NNP), (,/SYS))

d. (CD, (DT))
A seasonally adjusted 332.000
(332.000/CD, (a/DT), (adjusted/JJ, (seasonally/RB)))

e. (CD, (RB))
1987 and early 1988
(1987/CD, (and/CC, (1988/CD, (early/RB))))

Although all of these cases are due to a mismatch between the SR Task dependency trees and the input expected by our realiser, they point to different input configurations requiring different modifications (rewritings) to ensure compatibility with the realiser. The structured information given by the suspicion tree provides a clear description of the main tree configurations that need to be rewritten to correct generation failures induced by infinitival verbs. We used these information to implement the rewrite rules required to resolve the identified mismatches.

Forms that are suspicious in some but not all contexts. The suspicion tree can also highlight forms that are suspicious in some but not all contexts. For instance, the right frontier of the suspicion tree in Figure 8 shows that the CD (cardinals) part of speech occurs in several suspicious forms namely, (IN, (CD)) (a preposition dominating a cardinal), (CD, (IN)) (a cardinal dominating a preposition), (CD, (CD)) (a cardinal dominating a cardinal), (CD, (DT)) (a cardinal dominating a determiner) and (CD, (RB)) (a cardinal dominating an adverb). Examples for these configurations and their subcases are given in (2).

Noticeably, the suspicious form (CD) does not appear in the suspicion tree. In other words, the tree highlights the fact that cardinals lead to generation failure in the contexts shown but not in all contexts. Indeed, in this case, all suspicious forms points to a single cause of failure namely, a mis-

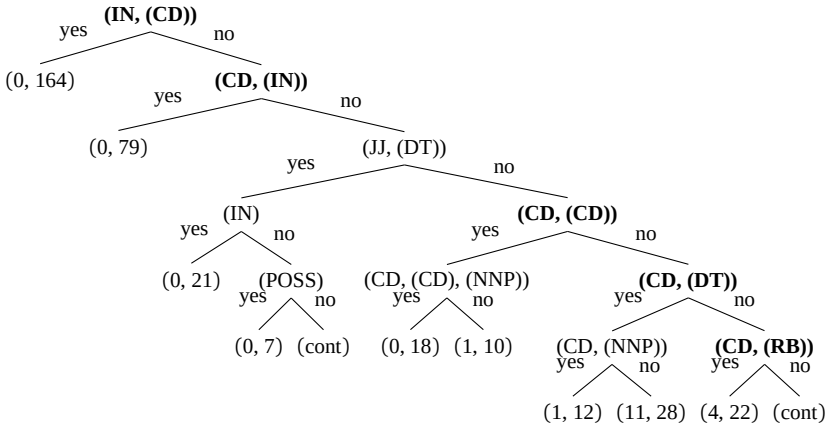


Figure 8: Suspicion Tree (POS) for Generation from the NP-6 data (after fixing genitive, coordination and determiner cases).

match between grammar and lexicon. In the TAG grammar used, the constructions illustrated in (2) all expect cardinals to be categorised as nouns. In the lexicon however, cardinals are categorised as determiners. We modified the lexicon to categorise cardinals as determiners, nouns and adjectives and rerun the generator on the input. In the newly built suspicion trees, cardinals no longer induce high generation failure rates. The fact that cardinals are not always associated with failure can be traced back to the fact that they are often used as determiners and that for this context, the lexicon contains the appropriate information.

3.3 Using Error Mining to Improve Generation Results

We now briefly report on how the suspicion tree algorithm can help improve a generation system by showing the impact of corrections on undergeneration.

Generating NPs. Table 1 summarises a run with 6 iterations between error mining and error correction on the NP data. The corrections involve rewriting the SR data to the format expected by our realiser, grammar corrections and lexicon enrichment. Each time a correction is applied, the suspicion tree is recomputed thereby highlighting the next most likely sources of errors. G(Coord) indicates a fix in the grammar for coordination (discussed in Section 3.2.1). R(Gen) involves rewriting dependency trees for genitive NPs (e.g., *Oakland 's thief*) (Section 3.2.1) and R(Dt) rewriting dependency trees with determiners to map its part-of speech from determiner (DT) to pronoun (PRP) (Section 3.2.2) and to noun (NN) (nominal positions, e.g., *That's good*). L(days) involves updating the lexicon with correct days/NN to TAG families mapping (Section 3.2.2). R(Adv) involves rewriting dependency trees with adverbs to map its part-of speech from adverb (RB) to adjective (JJ) (e.g., *real estate alone*) (Section 3.2.1).

As the table shows, error mining permits decreasing undergeneration by 22.6, 25.6 and 8.7 points for NPs of size 4, 6 and ALL respectively. This suggests that simple NPs can be generated but that bigger NPs still cause undergeneration (8.2% and 13% of the cases respectively for NP-6 and NP-ALL) presumably because of more complex modifiers such as relative clauses, PPs and multiple determiners. Since in particular, relative clauses also appear in sentences, we proceeded to error mine the sentence data so as to provide more data for the error mining algorithm and therefore get a more global picture of the most important causes of failure.

	Input Data	Init Fail	G(Coord)	R(Gen)	R(Dt)	L(days)	R(Adv)
NP-4	23468	5939 (*25.3*)	4246 (*18.1*)	999 (*4.3*)	833 (*3.6*)	678 (*2.9*)	649 (*2.7*)
NP-6	10520	3560 (*33.8*)	2166 (*20.6*)	956 (*9.1*)	881 (*8.4*)	876 (*8.3*)	865 (*8.2*)
NP-ALL	123523	26769 (*21.7*)	21525	16702	16263	16094	16028 (*13*)

Table 1: Diminishing the number of errors using information from error mining on NP data. The first column indicates the type of NP chunks to be processed, the second (*Input Data*) the number of NP chunks to be processed, the third (*Init Fail*) the number of input on which generation initially fails and the last 5 ones the decrease in errors (the number of failed cases with the *percentage failure*) after fixing error cases identified by the suspicion tree. R(X) indicates that the correction is obtained by rewriting the input for phenomena X, G(X) indicates corrections in the grammar and L(X) indicates corrections in the lexicon.

Generating Sentences. Tables 2 show the impact of corrections on generation for sentences. For this data, we start with all the improvements made during error mining on the NP data. Table 2 represents this step as *NP-Final* summarizing generation results after all improvements from Table 1. During error mining on the S data, *infinitival verbs* (discussed in Section 3.2.2) and *auxiliary verbs* appear as prominent mismatches between the SR dependency trees and the input expected by our generator. R(Inf) in Table 2 involves 3 different rewriting rules corresponding to dependency relations *im*, *oprd* and *prd* for rewriting dependency trees with infinitival verbs. R(Aux) indicates rewriting for dependency trees with Verb/Auxiliary nuclei (e.g., *the whole process might be reversed*).

	Input Data	NP-Final	R(Inf)	R(Aux)
S-6	3877	1707 (*44.0*)	753 (*19.4*)	398 (*10.3*)
S-8	3583	1749 (*48.8*)	936 (*26.1*)	576 (*16.1*)
S-ALL	26725	19280 (*72.1*)	17862 (*66.8*)	16445 (*61.5*)

Table 2: Diminishing the number of errors using information from error mining on S data. The first column indicates the type of sentences to be processed, the second (*Input Data*) the number of sentences to be processed, the third (*NP-Final*) the number of input (processed with all improvements from Table 1) on which generation fails and, the fourth and the fifth error rates after rewriting dependency trees for infinitival cases and auxiliary verb cases respectively.

Finally, Table 3 summarises results from Table 1 and Table 2 adding an extra final improvement step (*Final*) consisting of minor grammar improvement (trees for pre-determiner PDT added, e.g., *all these millions*), lexicon enrichment (mapping to TAG families corrected) and rewriting rule (mapping part-of-speech from conjunction CC to determiner DT, e.g., *neither/CC the Bush administration nor arms-control experts*). The "Final" row in this Table shows the impact of S error reduction on NP error reduction. As can be seen reducing S-errors positively impact NP errors throughout.

In total we defined 11 rewrite rules (Gen-1, Dt-4, Adv-1, Inf-3, Aux-1 and Final-1), made 2 grammar corrections and performed a few lexicon updates.

Coverage and accuracy. As the tables show, the corrections carried out after a few cycle of error mining and error correction helps achieve a large improvement in coverage for smaller dependency trees; we notice a large drop of 23.2 points (from 25.3% to 2.1%) in error rates for NP-4, 28.3 points for NP-6, 34.5 points for S-4 and 33.6 points for S-6. For larger dependency trees however, improvement is more limited and other error cases becomes visible. Thus, the failure rate is reduced

by 10.4 points for NP-ALL (NPs from minimum size 1 to maximum size 91 with the average size 4); and by 10.9 points for S-ALL (sentences from minimum size 1 to maximum size 134 with the average size 22). The suspicion tree built after the *Final* step shows that coordination cases appear as most suspicious forms. Although the corrections made for coordination in the grammar G(Coord) permit generating simple coordinations (e.g., *John and Mary likes beans.*, *We played on the roof and in the garden.*, *I cooked beans and she ate it.*), the grammar still fails to generate for more complex coordination phenomenon (e.g., verb coordination *I cooked and ate beans.*, gapping phenomenon *John eat fish and Harry chips.*, *I liked beans that Harry cooked and which Mary ate.*) (Sarkar and Joshi, 1996). Other top suspicious forms are multiword expressions (e.g., *at least*, *so far*) and foreign words (part-of-speech FW) (e.g., *the naczelnik*, *perestroika*, *product de jour*).

	NP-4	NP-6	NP-ALL	S-6	S-8	S-ALL
Input Data	23468	10520	123523	3877	3583	26725
Init Fail	5939 (*25.3*)	3560 (*33.8*)	26769 (*21.7*)	-	-	-
NP-Final	649 (*2.7*)	865 (*8.2*)	16028 (*13.0*)	1707 (*44.0*)	1749 (*48.8*)	19280 (*72.1*)
S-Final	-	-	-	398 (*10.3*)	576 (*16.1*)	16445 (*61.5*)
Final	503 (*2.1*)	584 (*5.5*)	13967 (*11.3*)	371 (*9.5*)	545 (*15.2*)	16374 (*61.2*)

Table 3: Overall impact of error mining on generation from different types of dependency trees. The first row indicates the type of dependency data to be processed and the second (*Input Data*) the number of data to be processed. The rows named (*Init Fail*), (*NP-Final*), (*S-Final*) and (*Final*) are initial error rates, errors after applying improvements from Table 1, errors after applying improvements from Table 2 and errors after final improvements respectively.

To assess the precision of the surface realiser after error mining, we computed the BLEU score for the covered sentence data and obtained a score of 0.835 for S-6, 0.80 for S-8 and 0.675 for S-ALL[4].

4 Conclusion

We introduced an error mining algorithm that takes inspiration from (Quinlan, 1986)'s ID3 algorithm to structure the output of error mining in a way that supports a linguistically meaningful error analysis. We demonstrated its workings by applying it to the analysis of undergeneration in a grammar based surface realisation algorithm. And we show that it permits quickly identifying the main sources of errors while providing a detailed description of the various subcases of these sources if any.

The approach is generic in that permits mining trees and strings for suspicious forms of arbitrary size and arbitrary conjunctions of labelling. It could be used for instance to detect and structure the n-grams that frequently induce parsing errors; or to identify subtrees that frequently occur in agrammatical output produced by a generator.

We are currently working on further improving the generator using the suspicion tree algorithm. In future work, we plan to use our error mining algorithm to detect the most likely sources of over-generation based on the output of a surface realiser; and to investigate whether the approach can be useful in automatically detecting treebank and parse errors (Boyd et al., 2008; Dickinson and Smith, 2011).

Acknowledgments The research presented in this paper was partially supported by the European Fund for Regional Development within the framework of the INTERREG IV A Allegro Project.

[4]The BLEU score before error mining and correction is not reported here since it has low coverage due to the mismatches between the structures provided by the SR task and those expected by the realiser.

References

Belz, A., White, M., Espinosa, D., Kow, E., Hogan, D., and Stent, A. (2011). The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France.

Boyd, A., Dickinson, M., and Meurers, D. (2008). On detecting errors in dependency treebanks. *Research on Language and Computation*, 6(2):113–137.

Chi, Y., Yang, Y., and Muntz, R. R. (2004). Hybridtreeminer: An efficient algorithm for mining frequent rooted trees and free trees using canonical form. In *Proceedings of the 16th International Conference on and Statistical Database Management (SSDBM)*, pages 11–20, Santorini Island, Greece. IEEE Computer Society.

de Kok, D., Ma, J., and van Noord, G. (2009). A generalized method for iterative error mining in parsing results. In *Proceedings of the 2009 Workshop on Grammar Engineering Across Frameworks (GEAF 2009)*, pages 71–79, Suntec, Singapore. Association for Computational Linguistics.

Dickinson, M. and Smith, A. (2011). Detecting dependency parse errors with minimal resources. In *Proceedings of the 12th International Conference on Parsing Technologies (IWPT 2011)*, Dublin, Ireland.

Gardent, C. and Narayan, S. (2012). Error mining on dependency trees. In *Proceedings of the 50th Meeting of the Association for Computational Linguistics (ACL)*, pages 592–600, Jeju, Korea.

Narayan, S. and Gardent, C. (2012). Structure-driven lexicalist generation. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, Mumbai, India.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, pages 81–106.

Rajkumar, R., Espinosa, D., and White, M. (2011). The osu system for surface realization at generation challenges 2011. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, pages 236–238, Nancy, France.

Sagot, B. and de la Clergerie, E. (2006). Error mining in parsing results. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 329–336, Sydney, Australia.

Sarkar, A. and Joshi, A. K. (1996). Coordination in tree adjoining grammars: Formalization and implementation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 610–615.

van Noord, G. (2004). Error mining for wide-coverage grammar engineering. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL)*, pages 446–453, Barcelona, Spain.